


```
CCCCCCCC 000000 NN NN VV VV BBBB BBBB RRRRRRRR
CCCCCCCC 000000 NN NN VV VV BBBB BBBB RRRRRRRR
CC 00 00 NN NN VV VV BB BB RR RR
CC 00 00 NN NN VV VV BB BB RR RR
CC 00 00 NNNN NN VV VV BB BB RR RR
CC 00 00 NNNN NN VV VV BB BB RR RR
CC 00 00 NN NN VV VV BBBB BBBB RRRRRRRR
CC 00 00 NN NN VV VV BBBB BBBB RRRRRRRR
CC 00 00 NN NN VV VV BB BB RR RR
CC 00 00 NN NN VV VV BB BB RR RR
CC 00 00 NN NN VV VV BB BB RR RR
CC 00 00 NN NN VV VV BB BB RR RR
CCCCCCCC 000000 NN NN VV VV BBBB BBBB RRRRRRRR
CCCCCCCC 000000 NN NN VV VV BBBB BBBB RRRRRRRR
RR RR
```

```
LL LL SSSSSSSS
LL LL SSSSSSSS
LL LL II II
LL LL II II
LL LL II II
LL LL II II
LL LL II II
LL LL II II
LL LL II II
LL LL II II
LL LL II II
LL LL II II
LLLLLLLLLL II II SSSSSSSS
LLLLLLLLLL II II SSSSSSSS
```



```
1 0001 0 MODULE CONVBR (
2 0002 0 IDENT = 'V04-000'
3 P 0003 0 %BLISS32[
4 P 0004 0 ADDRESSING_MODE(EXTERNAL=LONG_RELATIVE,NONEXTERNAL=LONG_RELATIVE)
5 0005 0 ]
6 0006 0 ) =
7 0007 1 BEGIN
8 0008 1
9 0009 1 *****
10 0010 1 *
11 0011 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
12 0012 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
13 0013 1 * ALL RIGHTS RESERVED.
14 0014 1 *
15 0015 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
16 0016 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
17 0017 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
18 0018 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
19 0019 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
20 0020 1 * TRANSFERRED.
21 0021 1 *
22 0022 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
23 0023 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
24 0024 1 * CORPORATION.
25 0025 1 *
26 0026 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
27 0027 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
28 0028 1 *
29 0029 1 *
30 0030 1 *****
31 0031 1
32 0032 1 ++
33 0033 1 FACILITY: DSR (Digital Standard RUNOFF) / DSRPLUS
34 0034 1
35 0035 1 ABSTRACT: Convert a binary number into a vector of roman numerals and
36 0036 1 return the result and character count.
37 0037 1
38 0038 1
39 0039 1 ENVIRONMENT: Transportable
40 0040 1
41 0041 1 AUTHOR: R.W.Friday CREATION DATE: April, 1979
42 0042 1
```


CONVBR
V04-000

Revision History

I 16
16-Sep-1984 00:11:02
14-Sep-1984 13:05:55

VAX-11 Bliss-32 V4.0-742
[RUNOFF.SRC]CONVBR.BLI;1

Page 2
(2)

:	44	0043	1	%SBTTL 'Revision History'
:	45	0044	1	
:	46	0045	1	MODIFIED BY:
:	47	0046	1	
:	48	0047	1	003 KFA00003 Ken Alden 07-Mar-1983
:	49	0048	1	Global edit of all modules. Updated module names, idents,
:	50	0049	1	copyright dates. Changed require files to BLISS library.
:	51	0050	1	
:	52	0051	1	--


```

99 0096 1 GLOBAL ROUTINE CONVBR (BINARY_NUMBER, KHARACTERS, KHA
100 0097 1
101 0098 1
102 0099 1 ++
103 0100 1 FUNCTIONAL DESCRIPTION:
104 0101 1
105 0102 1     Converts 'binary_number' to a vector of roman numerals,
106 0103 1     returning them in 'kharacters'; kharacter_count is the
107 0104 1     number of characters that result.
108 0105 1     The absolute value of 'binary_number' is converted,
109 0106 1     so that the user is responsible for handling negative numbers.
110 0107 1     ULM is as follows: -1 means return all characters in upper case.
111 0108 1     0 means return all characters in lower case.
112 0109 1     +1 means first character in upper case, rest in lower case.
113 0110 1
114 0111 1     NOTE
115 0112 1     The algorithm used here does not take into account many
116 0113 1     special cases for which shorter Roman forms are possible.
117 0114 1     For example, 45 is translated to XLV, even though VL is
118 0115 1     obviously shorter. This is not a bug, but is well within
119 0116 1     the tradition of how Roman numerals were formulated.
120 0117 1     The Romans themselves did not always use the "subtractive"
121 0118 1     principle in its fullest; it is possible to find Roman
122 0119 1     inscriptions that use IIXX for 18, for example. Similarly,
123 0120 1     VIIII is not uncommon for 9.
124 0121 1 FORMAL PARAMETERS:
125 0122 1
126 0123 1     See FUNCTIONAL DESCRIPTION
127 0124 1
128 0125 1 IMPLICIT INPUTS:      None
129 0126 1
130 0127 1 IMPLICIT OUTPUTS:     None
131 0128 1
132 0129 1 ROUTINE VALUE:
133 0130 1 COMPLETION CODES:      None
134 0131 1
135 0132 1 SIDE EFFECTS: None
136 0133 1
137 0134 1 --
138 0135 1
139 0136 2 BEGIN
140 0137 2
141 0138 2 BIND
142 0139 2     ROM_DIGITS = .KHARACTERS : VECTOR;
143 0140 2
144 0141 2 LOCAL
145 0142 2     DEC_DIGITS : VECTOR [20],
146 0143 2     DEC_DIG_COUNT,
147 0144 2     ROM_DIG_COUNT,
148 0145 2     T,
149 0146 2     X,
150 0147 2     XC;
151 0148 2
152 0149 2     !First get all the decimal digits separated out.
153 0150 2     DEC_DIG_COUNT = 0;           !Assume user supplied zero.
154 0151 2     ROM_DIG_COUNT = 0;         !....
155 0152 2
```



```
: 156      0153      2      !Force number into correct range.
: 157      0154      2      T = ABS(.BINARY_NUMBER) MOD 4000;
: 158      0155      2
: 159      0156      2      !Special case, if user supplied zero.  In such a case return '0'.
: 160      0157      2      IF .T EQL 0
: 161      0158      2      THEN
: 162      0159      2          BEGIN
: 163      0160      2              .KCHARACTER_COUNT = 1;
: 164      0161      2              ROM_DIGITS[0] = %C'0';
: 165      0162      2              RETURN
: 166      0163      2          END;
: 167      0164      2
: 168      0165      2      !And now do strip off the digits, one by one.
: 169      0166      2      WHILE (.T NEQ 0) DO
: 170      0167      2          BEGIN
: 171      0168      2              DEC_DIGITS [.DEC_DIG_COUNT] = .T MOD 10;
: 172      0169      2              T = .T/10;
: 173      0170      2              DEC_DIG_COUNT = .DEC_DIG_COUNT + 1;
: 174      0171      2          END;
: 175      0172      2
: 176      0173      2      !Convert decimal digits to roman numerals.
: 177      0174      2      DECR I FROM (.DEC_DIG_COUNT - 1) TO 0 DO
: 178      0175      2          BEGIN
: 179      0176      2              !Prior to converting the next decimal digit, do the equivalent
: 180      0177      2              !of multiplying the partial roman numeral result by 10.
: 181      0178      2              INCR J FROM 0 TO (.ROM_DIG_COUNT - 1) DO
: 182      0179      2                  ROM_DIGITS [.J] = (R10(.ROM_DIGITS [.J]));
: 183      0180      2
: 184      0181      2              !Now convert the next decimal digit.  This is done by
: 185      0182      2              !a simple table lookup, followed by copying into ROM_DIGITS.
: 186      0183      2              X = .ROM_TAB[.DEC_DIGITS[I]];          !Look up the roman equivalent of this digit.
: 187      0184      2              XC = CH$RCHAR_A(X);                  !Get the digit count into XC.
: 188      0185      2              !Discard zeroes (i.e., 10, 20, etc) but continue in the loop so
: 189      0186      2              !what's already been converted gets multiplied by 10.
: 190      0187      2              IF .DEC_DIGITS [.I] NEQ 0
: 191      0188      2              THEN
: 192      0189      2                  !Not zero, so convert it as usual
: 193      0190      2                  INCR J FROM (.ROM_DIG_COUNT + 1) TO (.ROM_DIG_COUNT + .XC) DO
: 194      0191      2                      BEGIN
: 195      0192      2                          ROM_DIGITS [.J - 1] = CH$RCHAR_A(X);
: 196      0193      2                          ROM_DIG_COUNT = .J;          !Update current length.
: 197      0194      2                      END;
: 198      0195      2              END;
: 199      0196      2
: 200      0197      2      !Set up the length for the user.
: 201      0198      2      .KCHARACTER_COUNT = .ROM_DIG_COUNT;
: 202      0199      2
: 203      0200      2      !Apply case conversion rules.
: 204      0201      2      IF .ULM EQL 0
: 205      0202      2      THEN
: 206      0203      2          !User is content with lower case, so just return.
: 207      0204      2          RETURN;
: 208      0205      2
: 209      0206      2      !Compute how many characters need to be converted to upper case.
: 210      0207      2      !The result is saved in T.
: 211      0208      2      IF .ULM EQL -1
: 212      0209      2      THEN
```



```
: 213      0210  2      T = .ROM_DIG_COUNT - 1      !Upper case
: 214      0211  2      ELSE
: 215      0212  2      T = 0;                      !Mixed case
: 216      0213  2
: 217      0214  2      !Now loop over the characters to be converted and make them upper case.
: 218      0215  2      INCR I FROM 0 TO T DO
: 219      0216  2      ROM_DIGITS [I] = .ROM_DIGITS [I] - %C'a' + %C'A';
: 220      0217  2
: 221      0218  2      RETURN;
: 222      0219  1      END;                        !End of CONVBR
```

```
.TITLE CONVBR
.IDENT \V04-000\

.PSECT $PLITS$,NOWRT,NOEXE,2

      00 00 30 01 00000 P.AAB: .ASCII <1>\0\<0><0>
      00 00 69 01 00004 P.AAC: .ASCII <1>\i\<0><0>
      00 69 69 02 00008 P.AAD: .ASCII <2>\ii\<0>
      69 69 69 03 0000C P.AAE: .ASCII <3>\iii\
      00 76 69 02 00010 P.AAF: .ASCII <2>\iv\<0>
      00 00 76 01 00014 P.AAG: .ASCII <1>\v\<0><0>
      00 69 76 02 00018 P.AAH: .ASCII <2>\vi\<0>
      69 69 76 03 0001C P.AAI: .ASCII <3>\vii\
      00 00 00 69 69 69 76 04 00020 P.AAJ: .ASCII <4>\viii\<0><0><0>
      00 78 69 02 00028 P.AAK: .ASCII <2>\ix\<0>
00000000' 00000000' 00000000' 00000000' 00000000' 00000000' 0002C P.AAA: .ADDRESS P.AAB, P.AAC, P.AAD, P.AAE, P.AAF, -
      00000000' 00000000' 00000000' 00000000' 00044 P.AAG, P.AAH, P.AAI, P.AAJ, P.AAK
```

ROM_TAB= P.AAA

```
.PSECT $CODE$,NOWRT,2

      00FC 00000
      5E B0 AE 9E 00002 .ENTRY CONVBR, Save R2,R3,R4,R5,R6,R7
      51 7C 00006 MOVAB -80(SP), SP
      50 04 AC D0 00008 CLRQ DEC DIG_COUNT
      50 03 18 0000C MOVL BINARY_NUMBER, R0
      50 50 CE 0000E BGEQ 1$
      7E 00 01 7A 00011 1$: EMUL #1, R0, #0, -(SP)
      54 54 8E 0000FA0 8F 7B 00016 EDIV #4000, (SP)+, T, T
      54 D5 0001F TSTL T
      09 12 00021 BNEQ 2$
      0C BC 01 D0 00023 MOVL #1, @KCHARACTER_COUNT
      08 BC 30 D0 00027 MOVL #48, @KCHARACTERS
      54 D5 0002B RET
      03 12 0002E 2$: TSTL T
      00A2 31 00030 BRW 15$
      7E 00 01 7A 00033 3$: EMUL #1, T, #0, -(SP)
      50 50 0A 7B 00038 EDIV #10, (SP)+, R0, R0
      6E41 54 0A D0 0003D MOVL R0, DEC_DIGITS[DEC_DIG_COUNT]
      54 0A C6 00041 DIVL2 #10, T
      51 D6 00044 INCL DEC_DIG_COUNT
      E4 11 00046 BRB 2$
```


Module Level Declarations

B 1
16-Sep-1984 00:11:02
14-Sep-1984 13:05:55VAX-11 Bliss-32 V4.0-742
[RUNOFF.SRC]CONVBR.BLI;1Page 7
(4)

	50		01	CE	00048	4\$:	MNEGL	#1, J		0178
			58	11	0004B		BRB	12\$		
	53	08	BC40	D0	0004D	5\$:	MOVL	@KCHARACTERS[J], R3		0179
00000069	8F		53	D1	00052		CMPL	R3, #105		
			06	12	00059		BNEQ	6\$		
	53	78	8F	9A	0005B		MOVZBL	#120, R3		
			3F	11	0005F		BRB	11\$		
00000078	8F		53	D1	00061	6\$:	CMPL	R3, #120		
			06	12	00068		BNEQ	7\$		
	53	63	8F	9A	0006A		MOVZBL	#99, R3		
			30	11	0006E		BRB	11\$		
00000063	8F		53	D1	00070	7\$:	CMPL	R3, #99		
			06	12	00077		BNEQ	8\$		
	53	6D	8F	9A	00079		MOVZBL	#109, R3		
			21	11	0007D		BRB	11\$		
00000076	8F		53	D1	0007F	8\$:	CMPL	R3, #118		
			06	12	00086		BNEQ	9\$		
	53	6C	8F	9A	00088		MOVZBL	#108, R3		
			12	11	0008C		BRB	11\$		
0000006C	8F		53	D1	0008E	9\$:	CMPL	R3, #108		
			06	12	00095		BNEQ	10\$		
	53	64	8F	9A	00097		MOVZBL	#100, R3		
			03	11	0009B		BRB	11\$		
	53		2A	D0	0009D	10\$:	MOVL	#42, R3		
A4	08	BC40	53	D0	000A0	11\$:	MOVL	R3, @KCHARACTERS[J]		
			50	F2	000A5	12\$:	AOBLSS	ROM_DIG_COUNT, J, 5\$		
			50	D0	000A9		MOVL	DEC_DIGITS[I], R0		0183
			56	D0	000AD		MOVL	ROM_TAB[R0], X		
			57	9A	000B5		MOVZBL	(X)+, XC		0184
			50	D5	000B8		TSTL	R0		0187
			19	13	000BA		BEQL	15\$		
55			57	C1	000BC		ADDL3	XC, ROM_DIG_COUNT, R5		0190
			52	D0	000C0		MOVL	ROM_DIG_COUNT, J		
			0C	11	000C3		BRB	14\$		
	53	08	BC40	DE	000C5	13\$:	MOVAL	@KCHARACTERS[J], R3		0192
	FC		86	9A	000CA		MOVZBL	(X)+, -4(R3)		
			50	D0	000CE		MOVL	J, ROM_DIG_COUNT		0193
F0			55	F3	000D1	14\$:	AOBLEQ	R5, J, -13\$		0190
			51	F4	000D5	15\$:	SOBGEQ	I, 16\$		0174
			03	11	000D8		BRB	17\$		
			FF6B	31	000DA	16\$:	BRW	4\$		
	0C	BC	52	D0	000DD	17\$:	MOVL	ROM_DIG_COUNT, @KCHARACTER_COUNT		0198
			10	AC	D5	000E1	TSTL	ULM		0201
			20	13	000E4		BEQL	22\$		
FFFFFFFF	8F	10	AC	D1	000E6		CMPL	ULM, #-1		0208
			06	12	000EE		BNEQ	18\$		
	54	FF	A2	9E	000F0		MOVAB	-1(R2), T		0210
			02	11	000F4		BRB	19\$		
			54	D4	000F6	18\$:	CLRL	T		0212
			01	CE	000F8	19\$:	MNEGL	#1, I		0215
			05	11	000FB		BRB	21\$		
	08	BC40	20	C2	000FD	20\$:	SUBL2	#32, @KCHARACTERS[I]		0216
F7			54	F3	00102	21\$:	AOBLEQ	T, I, 20\$		
			04	00106	22\$:	RET				0219

; Routine Size: 263 bytes, Routine Base: \$CODE\$ + 0000


```

: 223      0220 1
: 224      0221 1 END
: 225      0222 0 ELUDOM

```

```
!End of module
```

PSECT SUMMARY

Name	Bytes	Attributes
\$PLITS	84	NOVEC,NOWRT, RD ,NOEXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)
\$CODES	263	NOVEC,NOWRT, RD , EXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)

COMMAND QUALIFIERS

```
; BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:CONVBR/OBJ=OBJ$:CONVBR MSRC$:CONVBR/UPDATE=(ENH$:CONVBR)
```

```
; Size:          263 code + 84 data bytes
; Run Time:      00:05.2
; Elapsed Time:  00:14.5
; Lines/CPU Min: 2566
; Lexemes/CPU-Min: 16647
; Memory Used:   74 pages
; Compilation Complete
```


0338 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

0339

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY